Path Rewriting in Semistructured Data

Denis Debarbieux, Yves Roos, Sophie Tison, Yves André, Anne-Cécile Caron

Path Rewriting in Semistructured Data *

Denis Debarbieux, Yves Roos, Sophie Tison, Yves André, Anne-Cécile Caron
Laboratoire d'Informatique Fondamentale de Lille,
U.M.R. C.N.R.S. 8022
Université de Lille 1, 59655 Villeneuve d'Ascq Cedex. France.
e-mail:{debarbie, yroos, tison, andre, caronc }@lifl.fr

Abstract

We consider in this paper rooted labeled directed graphs as models for semistructured data. A path inclusion constraint, in such graphs, is an expression $p \preceq q$ where p and q are regular expressions over the alphabet of labels. An path inclusion constraint $p \preceq q$ is satisfied by a rooted labeled directed graph if the set of nodes reached, from the root, by the words of the language described by p is included in the set of nodes reached, from the root, by the words of the language described by q. We answer here to the following problem : given a set $\mathcal{C}$ of path inclusion constraints, given a regular expression p describing an infinite regular language, we want to compute, if it exists, a regular expression f describing a finite language, such that $p \preceq f$ for every graph satisfying $\mathcal{C}$.

1 Introduction

The notion of semistructured data comes from the Web. In semistructured databases, the data is unconstrained by any type system or schema and may have an irregular structure. It can be one or more XML documents, or a set of web pages with hyperlinks. In [1] the authors present some models and problematics about semistructured data. In this paper, we see a semistructured data as a rooted edge-labeled directed graph. In the following, we will also consider a semistructured data as a finite state automaton on words, even if this data is not a tool for recognizing languages.

The study of semistructured databases has generated the development of new query languages. Most of them are based on path expressions, which allow to reach nodes to arbitrary depth in the data graph. More precisely, a regular path expression is a regular expression on the alphabet of labels appearing in the data. Each path from the root to a node in the data is labeled by a word. The result of the query q, regular path expression, is the set of nodes reached from the root by a path labeled by a word of q.

Figure 1 gives an example of semistructured data. On this data, the result of the query `journal.(title + article.title)` is the set of nodes $\{11, 12, 13\}$.

One can express constraints on semistructured data. Some of these are called path constraints since they give restrictions on the data paths. Certain kinds of integrity constraints found in object-oriented databases and also common in semistructured databases can be expressed with path constraints. These constraints have been introduced by Abiteboul and Vianu in [2]. See [6], [10] or [4] which analyze different classes of path constraints. Here, we study path inclusion constraints. A path inclusion constraint is written $p \preceq q$

*This research was partially supported by Inria (MOSTRARE team).

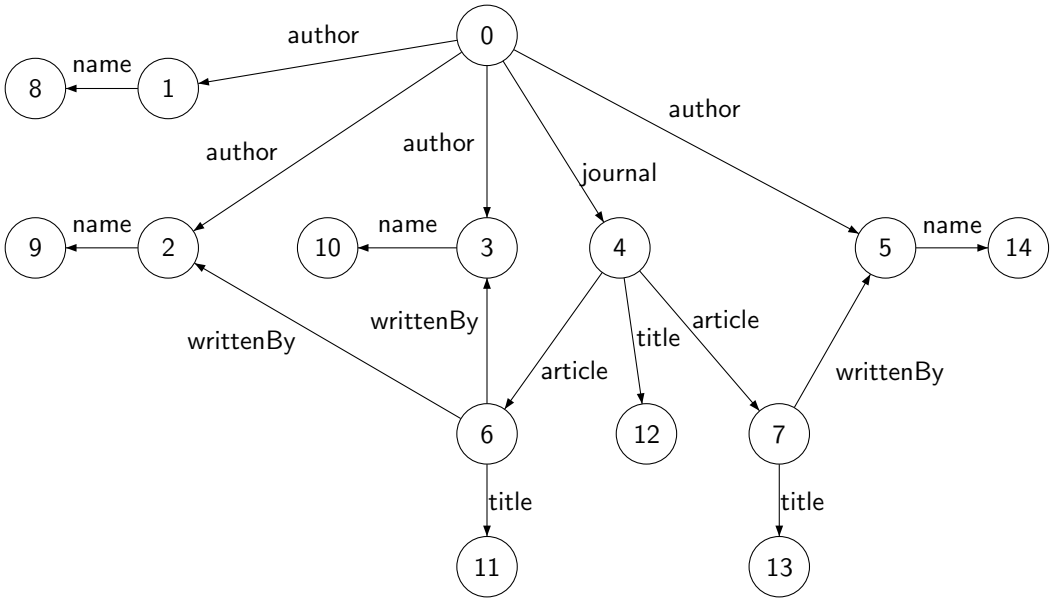


Figure 1: semistructured data

where p and q are regular path queries, and means that the set of nodes result of p is included in the set of nodes result of q. In Figure 1, we have `journal.article.writtenBy` $\preceq$ `author`. These constraints are interesting because they give semantic informations on the data (on the example, *an article in a journal is written by authors*) and they are used for query optimization.
In this context, the problem studied in this paper is the following : given a set $\mathcal{C}$ of inclusion constraints, given a regular expression q describing an infinite regular language, we want to compute, if it exists, a regular expression f corresponding to a finite language, such that $q \preceq f$ for every data satisfying $\mathcal{C}$. The existence of f is decidable when $\mathcal{C}$ contains only word inclusions [2] (inclusions $w_1 \preceq w_2$ where w_1 and w_2 are words) or inclusions of the form $p \preceq w$ where p is a regular expression and w is a word [5]. We define here a transducer computing f from q (when it exists) extending the result of [5].

2 Path inclusions and query rewriting

2.1 Semistructured data and path inclusions

In the sequel we use the following notions which were introduced in [2]. Let A be a fixed finite alphabet of labels.

Definition 2.1 : A semistructured datum is a triple $I = (N, i, T)$ where N is a set of nodes, $i \in N$ is called the root of datum I and $T \subseteq N \times A \times N$ is the set of transitions. If N is finite, the datum is said finite.

We are interested in the set of nodes which are reached by some paths in a datum. More precisely, given a datum $I = (N, i, T)$ and a word $u \in A^\star$, we denote by $u(I)$ the set of nodes that can be reached in I at the end of any path labelled by u starting from the root i, that is :

1. if $u = \varepsilon$, then $u(I) = \{i\}$

2. if $u = u'a$, with $u' \in A^\star$ and $a \in A$, $u(I) = \{n \in N \mid \exists n' \in u'(I), (n', a, n) \in T\}$

Then we can define the notion of query and result of query.

Definition 2.2 : A *regular query* p is a regular expression over A. The *result* of a query p over a datum I is the set $p(I) = \cup_{u \in L(p)} u(I)$ where $L(p)$ denotes the regular language described by p.

Definition 2.3 : A *path inclusion* is an expression of the form $p \preceq q$ where p, q are regular queries. A datum I *satisfies* a path inclusion $p \preceq q$, denoted $I \models p \preceq q$, if the set of nodes $p(I)$ is included in $q(I)$. I satisfies a set $\mathcal{C}$ of path inclusions, denoted $I \models \mathcal{C}$, if I satifies each path inclusion of $\mathcal{C}$.

Definition 2.4 : A set $\mathcal{C}$ of path inclusions *implies* a path inclusion $p \preceq q$, denoted $\mathcal{C} \models p \preceq q$, if for each datum I such that $I \models \mathcal{C}$, $I \models p \preceq q$.

We will use the following proposition [2]:

Proposition 2.1 *A set $\mathcal{C}$ of path inclusions implies a path inclusion $p \preceq q$, denoted $\mathcal{C} \models p \preceq q$, if for each **finite** datum I such that $I \models \mathcal{C}$, $I \models p \preceq q$.*

Definition 2.5 : A regular query p has the *boundedness property* w.r.t a set $\mathcal{C}$ of path inclusions if there exists a regular query f such that $\mathcal{C} \models p \preceq f$ and $L(f)$ is finite.

E.g., let $\mathcal{C}$ be $\{a^2 \preceq a\}$; w.r.t. $\mathcal{C}$, the query a^* is bounded, whereas the query $ba*$ is not.

2.2 Bounded path constraints and Prefix Rewriting

From now on, we will consider uniquely the case of a finite set of inclusions in the form $p \preceq u$ where p is a regular expression and u is a word: we shall call such path inclusions *bounded inclusions*. In this case, following and slightly generalizing [2], we associate with a set $\mathcal{C}$ of bounded path inclusions a rewriting system such that there is a prefix rewriting from u to v, if and only if the query u is included in the query v for each model of $\mathcal{C}$.

Definition 2.6 : Let $\mathcal{C} = \{p_1 \preceq u_1, p_2 \preceq u_2, \ldots, p_n \preceq u_n\}$ be a finite set of bounded path inclusions over an alphabet A. We consider the relation on words defined by $u \xrightarrow[\mathcal{C}]{} v$ if and only if there exists i such that $u \in L(p_i)$ and $v = u_i$. By extension, we denote also $\xrightarrow[\mathcal{C}]{}$ its rightcongruent closure. Then $\xrightarrow[\mathcal{C}]{\star}$ denotes the reflexive, transitive closure of $\xrightarrow[\mathcal{C}]{}$.

We can remark than this relation is a prefix rewriting relation as defined in [8] based on an infinite rewrite system.
From now on, we will suppose that there is no path constraint of the type $p \preceq \varepsilon$. Then we have the following property:

Proposition 2.2 *Let $\mathcal{C}$ be a set of bounded path inclusions. For any words u, v, $u \xrightarrow[\mathcal{C}]{\star} v$ if and only if $\mathcal{C} \models u \preceq v$.*

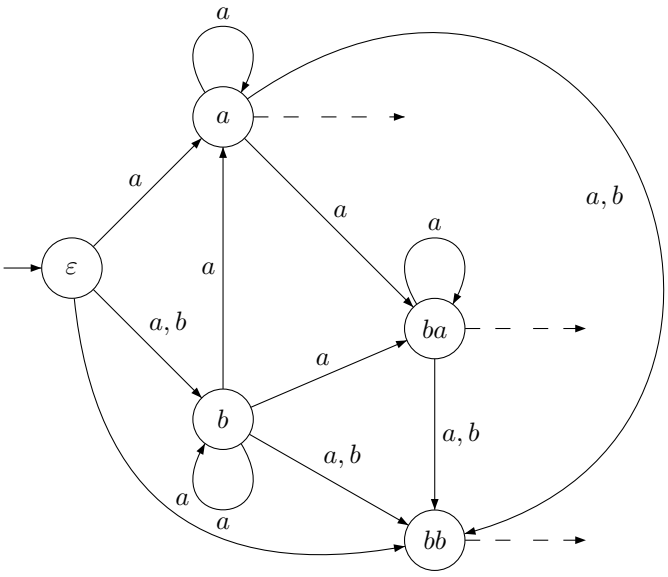
The proof uses a kind of (infinite) canonical model of $\mathcal{C}$ which is close to the model defined in [2]:

Definition 2.7 : With a set of bounded path constraints $\mathcal{C}$, we associate an infinite datum $I_{\mathcal{C}} = (N, i, T)$ defined by:

- $N = \{u \mid u \in A^\star\}$
- $i = \varepsilon$

- $T = \{(u, x, v) \mid v \xrightarrow[\mathcal{C}]{\star} ux\}$

Example 2.1 :
Let $\mathcal{C} = \{ab^\star \preceq ba, b^+ \preceq a, a(aa)^\star b \preceq a\}$. The following datum shows a finite part of the infinite datum defined in 2.7. It follows from $bb \to a \to ba \to aa$ that $(a, a, bb) \in T$, it follows from $b \to a \to ba$ that $(b, a, b) \in T$.



Then, we get immediately by induction on the length of u :

Lemma 2.1 $u(I_\mathcal{C}) = \{v \mid v \xrightarrow[\mathcal{C}]{\star} u\}$

Now, let us suppose $I_\mathcal{C} \models u \preceq v$, i.e. $u(I_\mathcal{C}) \subseteq v(I_\mathcal{C})$; by the preceding lemma u belongs to $u(I_\mathcal{C})$, so u belongs to $v(I_\mathcal{C})$; once again by the preceding lemma $u \xrightarrow[\mathcal{C}]{\star} v$. Let us suppose now $u \xrightarrow[\mathcal{C}]{\star} v$. So if $w \xrightarrow[\mathcal{C}]{\star} u$, $w \xrightarrow[\mathcal{C}]{\star} v$; by the preceding lemma $u(I_\mathcal{C}) \subseteq v(I_\mathcal{C})$, that is to say $I_\mathcal{C} \models u \preceq v$:

Lemma 2.2 *$I_\mathcal{C} \models u \preceq v$ if and only if $u \xrightarrow[\mathcal{C}]{\star} v$*

We obtain now

Lemma 2.3 *$I_\mathcal{C} \models u \preceq v$ if and only if $\mathcal{C} \models u \preceq v$*

Proof : First, it is easy to get than $I_\mathcal{C} \models \mathcal{C}$ and so that if $\mathcal{C} \models u \preceq v$, then $I_\mathcal{C} \models u \preceq v$. Now, let us suppose $I_\mathcal{C} \models u \preceq v$ that is to say $u \xrightarrow[\mathcal{C}]{\star} v$. Let $\preceq_\mathcal{C}$ be defined by $u \preceq_\mathcal{C} v$ if $\mathcal{C} \models u \preceq v$. The relation $\preceq_\mathcal{C}$ contains $\xrightarrow[\mathcal{C}]{}$, is transitive and closed by right-congruence: it contains $\xrightarrow[\mathcal{C}]{\star}$; so $u \xrightarrow[\mathcal{C}]{\star} v$ implies $u \preceq_\mathcal{C} v$ i.e. $\mathcal{C} \models u \preceq v$. □

By the two last lemmas, we obtain the proposition 2.2.
The following property will be fundamental:

Lemma 2.4 *If $I_\mathcal{C} \models u \preceq L$, there is some word v in L such that $I_\mathcal{C} \models u \preceq v$*

Proof : Indeed, if $I_{\mathcal{C}} \models u \preceq L$, $u(I_{\mathcal{C}})$ is included in $L(I_{\mathcal{C}})$; in particular, the state u belongs to $L(I_{\mathcal{C}})$, i.e. there is some v in L such that u belongs to $v(I_{\mathcal{C}})$; then, by lemma 2.1, $u(I_{\mathcal{C}})$ is included in $v(I_{\mathcal{C}})$, i.e. $I_{\mathcal{C}} \models u \preceq v$. □

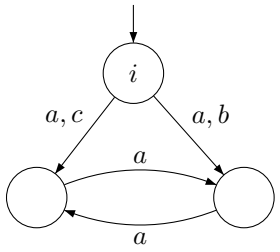
Then, by summarizing the preceding lemmas, we get:

Proposition 2.3 *Let $\mathcal{C}$ be a set of bounded path inclusions, and L a regular query; the following properties are equivalent:*

- $\mathcal{C} \models u \preceq L$
- *there is some word v in L such that $\mathcal{C} \models u \preceq v$*
- *there is some word v in L such that $u \xrightarrow[\mathcal{C}]{\star} v$*

Proof : If $\mathcal{C} \models u \preceq L$, as finite and infinite implication are equivalent and as $I_{\mathcal{C}} \models \mathcal{C}$, $I_{\mathcal{C}} \models u \preceq L$. So, there is some word v in L such that $I_{\mathcal{C}} \models u \preceq v$, and $\mathcal{C} \models u \preceq v$.
Let us now suppose $\mathcal{C} \models u \preceq v$, for some word v in L: $I_{\mathcal{C}} \models u \preceq v$ and then $u \xrightarrow[\mathcal{C}]{\star} v$.
Lastly, let us suppose $u \xrightarrow[\mathcal{C}]{\star} v$, for some word v in L: then $I_{\mathcal{C}} \models u \preceq v$. So, $\mathcal{C} \models u \preceq v$ and we get $\mathcal{C} \models u \preceq L$. □

The following example shows that proposition 2.3 does not hold in the general case of path inclusions which are not bounded inclusions :
Example 2.2 : The following datum satisfies $a^+ \preceq (b + c)$ but does not satisfy $a \preceq b$ neither $a \preceq c$.



2.3 Path inclusions and recognizable relations

In this section, we will use the notion of recognizable relations to decide boundedness of a regular query. The idea behind recognizable relations is simply to encode a n-uple of words on the alphabet A by a word on the alphabet $A \cup \{\bot\} \times \ldots \times A \cup \{\bot\}$, where $\bot$ is a new symbol. This encoding is obtained by superposing the n words, aligning the words by the end. E.g. (ab, aaa, b) will be encoded in $[\bot, a, \bot][a, a, \bot], [b, a, b]$. Formally, we define the morphism π_j from $([A \cup \{\bot\}]^n)^*$ by $\pi_j[a_1, a_2, ...a_n] = a_j$, if it belongs to A, $\pi_j[a_1, a_2, ...a_n] = \epsilon$ if $a_j = \bot$. Now, let Cor be the recognizable language of the words $u_1...u_n$ -on the alphabet $[A \cup \{\bot\}]^n$- satisfying the two following properties:

- for each j, if $\pi_j(u_k)$ is in A, $\pi_j(u_l)$ is in A for any $l > k$
- there exists some j s.t. $\pi_j(u_1)$ is in A.

Then, the notion of recognizable relation is defined by:
Definition 2.8 : A n-ary relation R on $A^\star \times \ldots \times A^\star$ is recognizable if and only if there exists an automaton $\mathcal{M}$ on the alphabet $A \cup \{\bot\} \times \ldots \times A \cup \{\bot\}$ recognizing the language

$$L(\mathcal{M}) = \{u \in Cor \ / \ (\pi_1(u), \cdots, \pi_n(u)) \in R\}$$

This definition corresponds to the definition of tree recognizable relations defined in [9], if we consider here a word as a tree whose root is associated with the end of the word. So, if *Rec* denotes the set of recognizable relations, the following properties of *Rec* (see [9] for more details) hold:

Proposition 2.4

- *Rec is closed under boolean operations (union, intersection, complementation)*
- *Rec is closed under cylindrification and projection.*[1]
- *Finiteness and emptiness are decidable on Rec.*

Proposition 2.5 (see for instance [9]) *If $\mathcal{C}$ is a set of inclusion constraints, the prefix rewriting relation $\xrightarrow[\mathcal{C}]{\star}$ is a recognizable relation.*

The boundedness property can be expressed in terms of recognizable relations. More precisely, let define some relations on $A^\star \times A^\star$:

- shorter $= \{(u,v) \mid |u| < |v|\}$
- shortestResults $= \{(u,v) \mid u \xrightarrow[\mathcal{C}]{\star} v \wedge (\nexists v' u \xrightarrow[\mathcal{C}]{\star} v' \wedge (v',v) \in \text{shorter})\}$

Lemma 2.5 *The relations shorter and shortestResults are recognizable.*

Proof : shorter is obviously a recognizable relation. Since $\xrightarrow[\mathcal{C}]{\star}$ is a recognizable relation, we can deduce from proposition 2.4 than shortestResults is a recognizable relation.
Let us now study the number of states of an automaton M for the relation shortestResults w.r.t. a set of bounded path inclusions $\mathcal{C} = \{p_1 \preceq u_1, p_2 \preceq u_2, \ldots, p_n \preceq u_n\}$, where the size of $\mathcal{C}$ is defined by $|\mathcal{C}| = \Sigma_{i=1}^{n} |p_i| + |u_i|$. We can define an automaton M_s of constant size for shorter(u,v) and an automaton M_c for $u \xrightarrow[\mathcal{C}]{\star} v$ of size $O(|\mathcal{C}|)$. Formula $\varphi = u \xrightarrow[\mathcal{C}]{\star} v' \wedge \text{shorter}(v',v)$ can so be associated with an automaton M_φ of size $O(|\mathcal{C}|)$. The automaton M_{sr} recognizing the relation shortestResults(u,v) can be built from a cartesian product of M_c and an automaton built from M_φ using a projection and a determinization: this leads to a size $O(2^{k|\mathcal{C}|})$ for some constant k. So, the number of states of the final automaton will be in $O(|\mathcal{C}|.2^{k|\mathcal{C}|})$. □

Proposition 2.6 *Let $\mathcal{C}$ a set of path inclusions and p a regular path query. Query p has the boundedness property w.r.t. $\mathcal{C}$ if and only if the recognizable set shortestResults $\cap$ $\{(u,v), u \in p\}$ is finite. So, the boundedness property of p is decidable.*

Let us note, that we could use a total recognizable ordering on words to associate with a word one and only one canonical query.
In the following, we develop an "ad hoc" method in order to reduce the number of states.

3 Query transduction

We consider here a fixed (non empty) finite set of bounded inclusions $\mathcal{C} = \{p_1 \preceq u_1, p_2 \preceq u_2, \ldots, p_n \preceq u_n\}$ over an alphabet A . The aim of this section is to build a finite word transducer $\tau_\mathcal{C}$ such that for any regular query p over A :

1. $\mathcal{C} \models p \preceq q$ for all q such that $L(q) = \tau_\mathcal{C}(L(p))$
2. p has the boundedness property w.r.t. $\mathcal{C}$ if and only if $\tau_\mathcal{C}(L(p))$ is finite

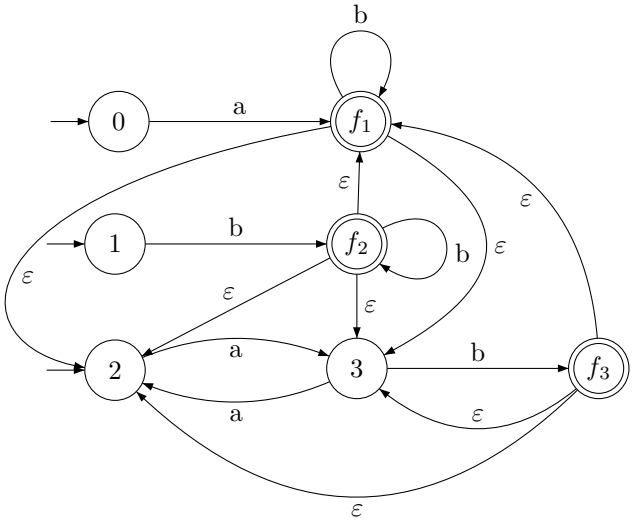
[1] roughly speaking, cylindrification (projection) consists in adding (deleting) a component in a tuple.

In order to define our transducer $\tau_{\mathcal{C}}$, our first goal is to build a finite automaton $\mathcal{A}_{\mathcal{C}}$ (with ε-moves) which recognizes the language $R_{\mathcal{C}} = \{v \in A^\star \mid \exists i, v \xrightarrow[\mathcal{C}]{+} u_i\}$. It is already known that $R_{\mathcal{C}}$ is a recognizable language from [7], [9], [8]. We give here a different construction : For each i with $1 \leq i \leq n$, let $\mathcal{M}_i = (A, Q_i, I_i, F_i, \delta_i)$ be an automaton recognizing the language $L(p_i)$. We can assume, without loss of generality, that for different subscripts i and j, the intersection $Q_i \cap Q_j$ is empty. Then we can define $\mathcal{A}_{\mathcal{C}} = (A, Q, I, F, \Delta)$ where $Q = \cup_{i=1}^n Q_i$, $I = \cup_{i=1}^n I_i$, $F = \cup_{i=1}^n F_i$ and $\Delta = \cup_{k\in\mathbb{N}} \Delta_k$ where Δ_k, for $k \in \mathbb{N}$ is defined inductively by :

- $\Delta_0 = \cup_{i=1}^n \delta_i$
- for $k > 0$, $\Delta_k = \Delta_{k-1} \cup \{(q, \varepsilon, q') \mid \exists i \leq n, q \in F_i, q' \in \Delta_{k-1}(I, u_i)\}$

Since Δ is included in $Q \times (A \cup \{\varepsilon\}) \times Q$, it is clear that there exists an integer K such that $\Delta_K = \Delta_{K+1} = \Delta$. Since we have $K \leq |Q|^2$, automaton $\mathcal{A}_{\mathcal{C}}$ can be build in quadratic time in $|\mathcal{C}|$, the size of $\mathcal{C}$.

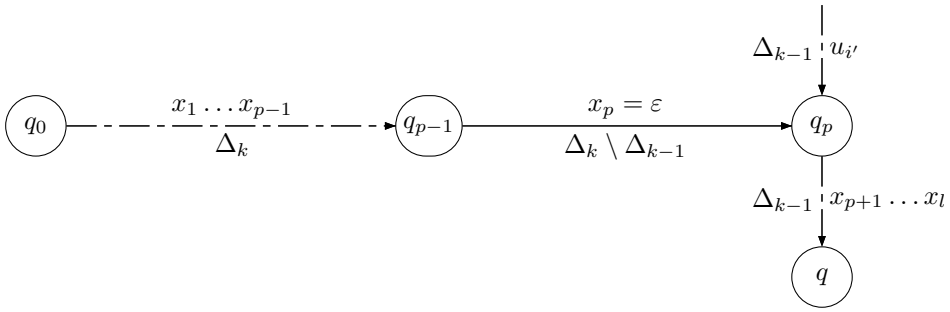
Example 3.1 : Let $\mathcal{C} = \{ab^\star \preceq ba, b^+ \preceq a, a(aa)^\star b \preceq a\}$. Automaton $\mathcal{A}_{\mathcal{C}}$ is the following :



We have now to prove that $\mathcal{A}_{\mathcal{C}}$ recognizes $R_{\mathcal{C}}$.

Lemma 3.1 *For any word v in $A^\star$, if $\Delta(I, v) \cap F_i \neq \emptyset$ for some i in $\{1, \ldots, n\}$ then $v \xrightarrow[\mathcal{C}]{+} u_i$.*

Proof : Let $v \in A^\star$ and $q \in \Delta(I, v) \cap F_i$ for some i in $\{1, \ldots, n\}$. Then there exists a k such that $q \in \Delta_k(I, v)$. We will show that $v \xrightarrow[\mathcal{C}]{+} u_i$ by induction on k. If $k = 0$, then $v \in L(p_i)$ and $v \xrightarrow[\mathcal{C}]{} u_i$. Suppose now that $k > 0$. There exist $q_0, q_1, \ldots, q_l$ in Q and $x_1, x_2, \ldots, x_l$ in $A \cup \{\varepsilon\}$ such that $q = q_l$, $v = x_1 x_2 \ldots x_l$ and for any j in $\{1, \ldots, l\}$, $(q_{j-1}, x_j, q_j) \in \Delta_k$. Let m be the number of such (q_{j-1}, x_j, q_j) which are in $\Delta_k \setminus \Delta_{k-1}$. We shall now make an induction on m. If $m = 0$, then, by induction hypothesis, we obtain that $v \xrightarrow[\mathcal{C}]{+} u_i$. If $m > 0$, let p be the integer such that $(q_{p-1}, x_p, q_p) \in \Delta_k \setminus \Delta_{k-1}$ and for any j with $p < j \leq l$, (q_{j-1}, x_j, q_j) is in Δ_{k-1}. Then $x_p = \varepsilon$, $q_{p-1} \in F_{i'}$ for some i' in $\{1, \ldots, n\}$ and $q_p \in \Delta_{k-1}(I, u_{i'})$:



By induction hypothesis on m, we obtain that $x_1 x_2 \dots x_{p-1} \xrightarrow[\mathcal{C}]{+} u_{i'}$ and by induction hypothesis on k, we obtain that $u_{i'} x_p x_{p+1} \dots x_l \xrightarrow[\mathcal{C}]{+} u_i$. It follows that $v = x_1 x_2 \dots x_l \xrightarrow[\mathcal{C}]{+} u_{i'} x_p x_{p+1} \dots x_l \xrightarrow[\mathcal{C}]{+} u_i$. □

In order to prove the converse of lemma 3.1, we shall use the following result :

Lemma 3.2 *Let v and w be two words of $A^\star$, then $v \xrightarrow[\mathcal{C}]{\star} w \Longrightarrow \Delta(I, w) \subseteq \Delta(I, v)$.*

Proof : Let j be the length of the derivation $v \xrightarrow[\mathcal{C}]{\star} w$. We shall make an induction on j. If $j = 0$ then $v = w$ and $\Delta(I, w) = \Delta(I, v)$. If $j > 0$, then there exist i in $\{1, \dots, n\}$ and words v_1, v_2 such that $v \xrightarrow[\mathcal{C}]{j-1} v_1 v_2$, $v_1 \xrightarrow[\mathcal{C}]{} u_i$ and $w = u_i v_2$. By induction hypothesis, we have $\Delta(I, v_1 v_2) \subseteq \Delta(I, v)$. Moreover, since $v_1 \xrightarrow[\mathcal{C}]{} u_i$ then $v_1 \in L(p_i)$ and there exists a state q in $F_i \cap \Delta(I, v_1)$. Let q' be a state in $\Delta(I, u_i)$ then there exists an integer k such that $q' \in \Delta_k(I, u_i)$. It follows that $(q, \varepsilon, q') \in \Delta_{k+1} \subseteq \Delta$ and $q' \in \Delta(I, v_1)$. As we have $\Delta(I, u_i) \subseteq \Delta(I, v_1)$, we obtain $\Delta(I, w) = \Delta(I, u_i v_2) \subseteq \Delta(I, v_1 v_2) \subseteq \Delta(I, v)$. □

We are now able to prove :

Proposition 3.1 *For any word v in $A^\star$, $\Delta(I, v) \cap F_i \neq \emptyset$ for some i in $\{1, \dots, n\}$ if and only if $v \xrightarrow[\mathcal{C}]{+} u_i$. Then automaton $\mathcal{A}_\mathcal{C}$ recognizes $R_\mathcal{C}$*

Proof : From lemma 3.1, we have only to prove that for any word $v \in A^\star$, if there exists an i in $\{1, \dots, n\}$ such that $v \xrightarrow[\mathcal{C}]{+} u_i$ then $\Delta(I, v) \cap F_i \neq \emptyset$. Since $v \xrightarrow[\mathcal{C}]{+} u_i$, there exists a word $w \in A^\star$ such that $v \xrightarrow[\mathcal{C}]{\star} w \xrightarrow[\mathcal{C}]{} u_i$. Then $w \in L(p_i)$ and $\Delta(I, w) \cap F_i \neq \emptyset$. Since $\Delta(I, w) \subseteq \Delta(I, v)$, from lemma 3.2 it follows that $\Delta(I, v) \cap F_i \neq \emptyset$. □

Now, we construct the transducer $\tau_\mathcal{C}$ from the automaton $\mathcal{A}_\mathcal{C}$. This transducer produces from any regular query p which has the boundedness property w.r.t. $\mathcal{C}$ a regular query q, such that $L(q)$ is finite and $\mathcal{C} \models p \preceq q$.

Definition 3.1 : Let $\mathcal{C}$ be a set of bounded inclusions over A and v be a word of $A^\star$. The shortest suffix of v which cannot be rewritten w.r.t $\xrightarrow[\mathcal{C}]{\star}$, denoted by $f_\mathcal{C}(v)$ is the suffix of v which satisfies that for any words $v_1 \in R_\mathcal{C} \cup \{\varepsilon\}$ and $v_2 \in A^\star$, if $v = v_1 v_2$ then $|v_2| \geq |f_\mathcal{C}(v)|$.

It is easy to verify that the following lemma holds :

Lemma 3.3 *Let $\mathcal{C}$ be a set of bounded inclusions over A. Let v and w be two words of $A^\star$. Then $v \xrightarrow[\mathcal{C}]{\star} w$ if and only if there exist v_1 and w_1 in $A^\star$ such that $v = v_1 f_\mathcal{C}(v)$, $w = w_1 f_\mathcal{C}(v)$ and $v_1 \xrightarrow[\mathcal{C}]{\star} w_1$.*

The shortest suffix of words which cannot be rewritten w.r.t $\xrightarrow[\mathcal{C}]{\star}$ has been used to decide if a regular query has the boundedness property w.r.t. a set of bounded inclusions.

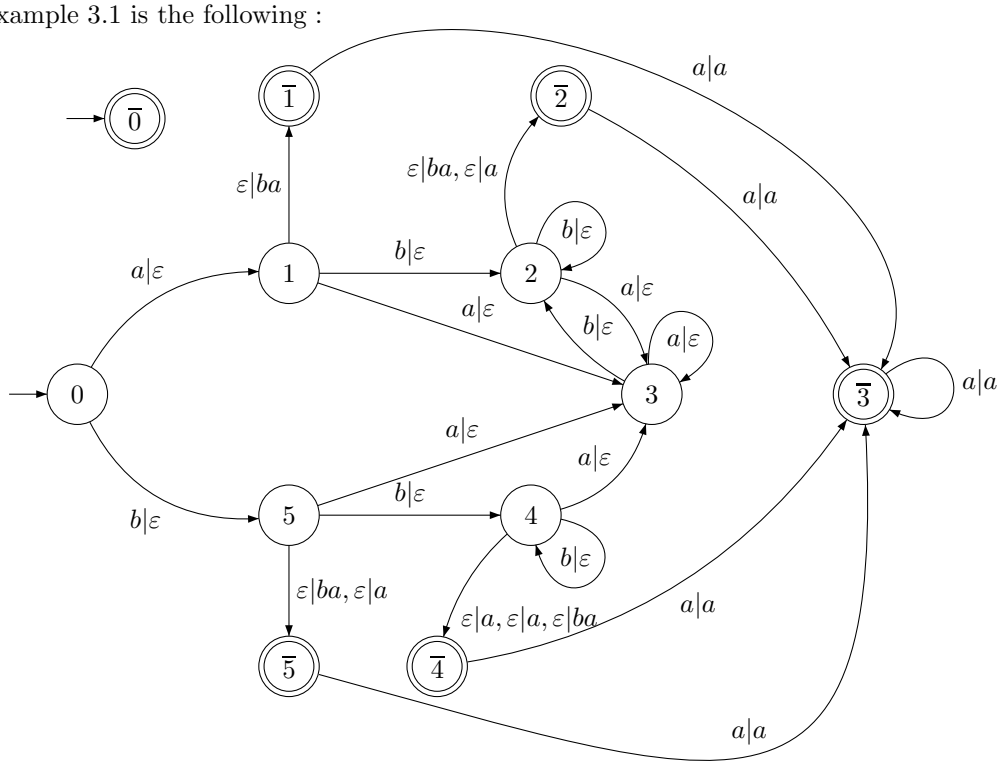
Theorem 3.1 ([5]) *Let $\mathcal{C}$ be a set of bounded inclusions over A and p be a regular query over A then p has the boundedness property w.r.t. $\mathcal{C}$ if and only if the set $F_{\mathcal{C}}(p) = \{f(v) \mid v \in L(p)\}$ is finite.*

Our transducer $\tau_{\mathcal{C}}$ will satisfy the following property, proved in proposition 3.2 :
For any word $v \in A^\star$, we will have $w \in \tau_{\mathcal{C}}(v)$ if and only if $w = v = f(v)$ or $v \xrightarrow[\mathcal{C}]{+} w \wedge w = u_i f(v)$ for some $i \in \{1, \ldots, n\}$.

Let $\mathcal{A}'_{\mathcal{C}}$ be the complete and deterministic automaton obtained from the automaton $\mathcal{A}_{\mathcal{C}}$ applying the classical transformations (remove ε-moves, subset-construction, completion). Then $\mathcal{A}'_{\mathcal{C}} = (A, Q', q'_0, F', \Delta')$ and, from the subset-construction and proposition 3.1, we can associate to any state $q \in Q'$ a subset $S(q) \subseteq \{1, \ldots, n\}$ such that, for any word $v \in A^\star$, $i \in S(\Delta'(q'_0, v))$ if and only if $v \xrightarrow[\mathcal{C}]{+} u_i$. In particular, $F' = \{q \in Q' \mid S(q) \neq \emptyset\}$. We can remark than the automaton $\mathcal{A}'_{\mathcal{C}}$ can be built in time $2^{|C|}$.

Then the transducer $\tau_{\mathcal{C}}$ is defined by : $\tau_{\mathcal{C}} = (A, Q' \cup \bar{Q}', \{q_0, \bar{q_0}\}, \bar{Q}', T)$ where A is the input/output alphabet, $Q' \cup \bar{Q}'$ where $\bar{Q}'$ is a copy of Q' is the set of states, q_0 and $\bar{q_0}$ are the initial states, $\bar{Q}'$ is the set of final states and $T \subseteq Q \times A \cup \{\varepsilon\} \times A^\star \times Q$ is the set of transitions of $\tau_{\mathcal{C}}$ defined by : $T = \{(q, x, \varepsilon, q') \mid q, q' \in Q', x \in A, (q, x, q') \in \Delta'\}$
$\cup \{q, \varepsilon, u_i, \bar{q}) \mid q \in Q', i \in S(q)\}$
$\cup \{(\bar{q}, x, x, \bar{q'}) \mid q, q' \in Q', x \in A, S(q') = \emptyset, (q, x, q') \in \Delta'\}$.

Example 3.2 : The transducer $\tau_{\mathcal{C}}$ associated with the set of bounded inclusions of example 3.1 is the following :



Proposition 3.2 *For any words $v, w \in A^\star$, $w \in \tau_{\mathcal{C}}(v)$ if and only if $w = v = f(v)$ or $v \xrightarrow[\mathcal{C}]{+} w$ and $w = u_i f(v)$ for some i in $\{1, \ldots, n\}$.*

Proof :

- Let $v, w \in A^\star$ such that $w \in \tau_{\mathcal{C}}(v)$ then there exists a path in $\tau_{\mathcal{C}}$ from an initial state to a final state, labelled by v. If this path starts from $\bar{q}_0$, then $w = v$ and, since for any $(\bar{q}, x, x, \bar{q'})$ in T, $S(q') = \emptyset$, there does not exist any prefix of v, different from ε which belongs to $R_{\mathcal{C}}$. It follows that $f(v) = v = w$. Let us suppose now that the path labelled by v starts from q_0. Then there exists two words v_1 and v_2 such that $v = v_1 v_2$, v_1 labels a path from q_0 to a state $q \in Q'$ such that there exists some i in $S(q)$, $(q, \varepsilon, u_i, \bar{q})$ is in T, v_2 labels a path from $\bar{q}$ to some state $\bar{q'}$ of $\bar{Q'}$. Then $v_1 \xrightarrow[\mathcal{C}]{+} u_i$, $w = u_i v_2$ and $v \xrightarrow[\mathcal{C}]{+} w$. Note that for any prefix v_2' of v_2, different from ε, we are sure that $v_1 v_2'$ does not belong to $R_{\mathcal{C}}$ since v_2' labels a path from $\bar{q}$ to some state $\bar{q''}$ with $S(q'') = \emptyset$. It follows that $v_2 = f(v)$ then $w = u_i f(v)$.
- Conversely, let $v, w \in A^\star$ such that $w = v = f(v)$ then there does not exist any prefix of v, different from ε which belongs to $R_{\mathcal{C}}$. Then v labels a path from $\bar{q_0}$ to some state $\bar{q'}$ of $\bar{Q'}$ and $v \in \tau_{\mathcal{C}}(v)$. Suppose now that $v \xrightarrow[\mathcal{C}]{+} w$ and $w = u_i f(v)$ for some i in $\{1, \ldots, n\}$. From lemma 3.3, it follows that $v = v_1 f(v)$ and $v_1 \xrightarrow[\mathcal{C}]{+} u_i$. Then v_1 labels a path from q_0 to a state q such that $i \in S(q)$. Moreover, it is clear that $f(v)$ labels a path from $\bar{q}$ to some state $\bar{q'}$ of $\bar{Q'}$. It follows that $u_i f(v) = w$ is in $\tau_{\mathcal{C}}(v)$.

□

Proposition 3.3 *Let $\mathcal{C} = \{p_1 \preceq u_1, p_2 \preceq u_2, \ldots, p_n \preceq u_n\}$ be a non-empty finite set of bounded inclusions over an alphabet A and let $\tau_{\mathcal{C}}$ be the transducer defined above, then for any regular query p over A :*

1. *$\mathcal{C} \models p \preceq q$ for all q such that $L(q) = \tau_{\mathcal{C}}(L(p))$*
2. *p has the boundedness property w.r.t. $\mathcal{C}$ if and only if $\tau_{\mathcal{C}}(L(p))$ is finite*

Proof :

1. Let u be a word of $L(p)$ and v be a word of $\tau_{\mathcal{C}}(u) = L(q)$ $(\forall u \in A^\star, \tau(u) \neq \emptyset)$. It follows from proposition 3.2 that $u \xrightarrow[\mathcal{C}]{+} v$ or $u = v$ and then $\mathcal{C} \models p \preceq q$ from proposition 2.3.
2. If $\tau_{\mathcal{C}}(L(p))$ is finite then p has clearly the boundedness property w.r.t $\mathcal{C}$. Conversely, if p has the boundedness property w.r.t. $\mathcal{C}$ then, from theorem 3.1, $F_p = \{f(w), w \in L(p)\}$ is a finite set. Since, from proposition 3.2, $\tau_{\mathcal{C}}(L(p)) \subseteq \{\varepsilon, u_1, ..., u_n\} F_p$ it follows $\tau_{\mathcal{C}}(L(p))$ is finite.

□

Remark : We have produced an algorithm which, given a non-empty finite set of bounded inclusions $\mathcal{C}$ and a regular query p gives a finite query f such that $\mathcal{C} \models p \preceq f$ if and only if p has the boundedness property w.r.t. $\mathcal{C}$. This algorithm is in $O(|p|.2^{|C|})$ since it computes $\tau_{\mathcal{C}}(L(p))$. It is worth noting that the problem of deciding boundedness property is PSPACE-hard : Indeed let p and q be two regular expressions over an alphabet A. Let us consider a new letter \$ which is not in A and let us define $\mathcal{C}$ as $\mathcal{C} = \{q\$^+ \preceq \$\}$. Then it is easy to see that p has the boundedness property for $\mathcal{C}$ if and only if $L(p) \subseteq L(q)$. It follows that our algorithm may be used to decide whether a regular language is included in another regular language where these languages are defined by regular expressions and

it is known from [3] that this last decision problem is PSPACE-complete.

When the set of path inclusions $\mathcal{C}$ contains not only bounded inclusions, the problem to check whether a regular query p has the boundedness property w.r.t. $\mathcal{C}$ is still open. It is also an open question to know whether this problem is decidable or not.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann Publishers, 2000.

[2] Serge Abiteboul and Victor Vianu. Regular path queries with constraints. In *PODS*, pages 122–133. ACM Press, 1997.

[3] A. Aho, J. Hopcroft, and J. Ullman. *The design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Compagny, Reading, Mass., 1974.

[4] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13(6):939 – 956, 2003.

[5] Y. André, F. Bossut, and A.C. Caron. On decidability of boundedness property for regular path queries. In *proceedings of DLT'99*, Aachen, Germany, 1999. Development in Language Theory, World Scientific Publishing Co.

[6] P. Buneman, W. Fan, and S. Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2), 2000.

[7] J. Richard Büchi and W.H. Hosken. Canonical systems which produce periodic sets. *Mathematical Systems Theory*, 4(1), 1970.

[8] D. Caucal. On the regular structure of prefix rewritings. In Springer, editor, *Selected papers of the conference on Fifteenth colloquium on trees in algebra and programming*, pages 87 – 102, Copenhagen, Denmark, May 1990.

[9] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.

[10] G. Grahne and A. Thomo. Query containment and rewriting using views for regular path queries under constraints. In *proceedings of PODS'03*, pages 111–122. Symposium on Principles of Database Systems, ACM, 2003.